Commodore

comments and bulletins concerning your COMMODORE PET tm

The Transactor

Vol. 2
BULLETIN # 2

PET tm is a registered trademark of Commodore Business Machines Inc.

June 30, '79

PRINT Speed - Up

For those who have always wished they could get information displayed on the screen faster without having to go to machine language, the answer has been found by Larry Phillips of the Vancouver PET User Group. On most 8K and, so far, all 16/32K PETs, POKE 59458,62 (normally 30) speeds up vertical PRINTing by almost double and horizontal PRINTing by a factor of almost 4!

How It Works

All TV picture screens are activated by an electron beam. This beam starts at the top left of the screen and sweeps left to right across every row of dots right to the bottom of the screen. In actual fact it sweeps the odd rows first (1,3,5,...199) and then returns to sweep the even rows (2,4,6,...200). This function is accomplished by the Video Sweep Generator (VSG) and is totally independent of BASIC. The VSG must also perform one other important function and that is, as mentioned previously, return the beam from the bottom right of the screen to the top left. This is known as vertical retrace or simply retrace. When in retrace the electron beam is turned OFF else a diagonal streak would appear as the beam returns. Once there, the beam is turned back ON and begins sweeping again. When the beam is sweeping or 'writing' the screen is continuously accessing each consecutive byte of screen memory and echoing the information to the screen. During write, the VSG signals PET and suggests that the 6502 not change the contents of screen memory. This prevents "snow". The only time that the 6502 should access screen memory is during retrace; when the beam is totally OFF and cannot possibly cause snow. When retrace occurs, the VSG signals the 6502 and in the short time it takes for the beam to return, the 6502 can alter screen memory if necessary (i.e. upon execution of a PRINT statement). It may get 2 or 3 characters in before the beam goes back into write and the 6502 halts any alteration.

Commodore Systems. 3370 Pharmacy Avenue, Agincourt, Ont.(416)499-4292

But what if we make the PET 'think' that it's always in vertical retrace; then it would alter screen memory anytime instead of waiting for retrace. The display may sustain a little snow but the greatly increased speed might make up for it. You decide but here's how it's done. The VSG actually signals the 6522 which relays the electron beam status to the 6502; either ON or OFF. The signal comes from the VSG and enters the PB port of the 6522 at pin 5 (PB5) and is stored in Data Register B (DRB) or decimal location 59456. PB5 is configured as an input on power up by the 6522's Data Direction Register B (DDRB) at decimal 59458. When in write the VSG sends a logic 1 to the input line PB5 and in retrace a logic 0 is sent. However, if the 6502 is prevented from ever seeing that logic 1, it will never even suspect that the screen is in write and will slam characters into screen memory without hesitation. Of course we can't disconnect the line but with software we can make it an output and ground it by placing a logic 0 in the appropriate bit. On power up DDRB (59458) contains 30 which means bit 5 contains 0 dictating that PB5 is an input line. If 32 is added to the contents of DDRB, PB5 is now configured as an output. So if 59458 is POKEd with 62, and 59456 is POKEd with 223 (255-32), PET now constantly receives 0 from the output line PB5 indicating vertical retrace.

Of course the screen is not always in vertical retrace, PET just 'thinks' it is and that's all we need. This procedure, for some reason, doesn't work on every 8K PET. To be absolutely sure it will work on yours, try the following immediatly after power up:

```
POKE 59456,223 : POKE 59458,62
```

If it works the first POKE may be optional. Now try the following program and note the difference in speed!

```
10 PRINT "cs";
20 POKE 59458,30
30 FOR J = 1 TO 400
40 PRINT "*";
50 NEXT : PRINT
60 POKE 59458,62
70 FOR J = 1 TO 400
80 PRINT "*";
90 NEXT : GOTO 10
```

BITS and PIECES

Some interesting discoveries have been unearthed recently for the 8K and 16/32K versions of the PET. The single most important one, I feel, was uncovered by who else but Jim Butterfield. Burried deep in the keyboard interrupt routines is some code which does a test for the "<" key. To see this amazing little feature operate, insert a tape into cassette #1 and simply press 'PLAY' (not a LOAD). Now hold down the "<" key and PET will tell you immediatly if there is something recorded on that tape. If there is, the "<" sign will repeat across the screen at the rate of about 5/sec. If not, no repetition will occur. Now we have a way to check tapes before recording something over material we may have wanted to keep and, more importantly, tapes can now be cued up to blank tape without having to load in the last program or file. Fantastic! The test works on all PETs but only for cassette #1.

CRASH Your PET!

The following is a list of rather interesting crashes for 8K PETs. They can cause absolutely no internal damage to the machine that power-down and up won't fix.

1. This one might make a good screen-alignment test:

```
Type:   10 ABC
 Now:   POKE 1025,0
Type:   10 DEF
```

2. Decimal location 537 (0219 hex) is the low order byte of the hardware interrupt vector. Try the following and also experiment...

```
POKE 537,49
POKE 537,50
```

3. On a clear screen in the 'HOME' position, type:

```
2 RVS field '*'s then RVS Off;
A shifted 'L';
An '@' sign;
A RVS '@' sign.
```

   The characters just typed should appear in the first 5 positions of the top line. Hit 'RETURN' and, of course, get a ?SYNTAX ERROR. Now SYS 32768. (SYS to the first location of screen memory) Change the display by holding down various combinations of keys (STOP, RETURN, etc.) The result result can be altered by varying the number of RVS '*'s on the top line.

4. On a clear screen in the 'HOME' position type a shifted closing bracket (■) and 'RETURN'. Now type:

```
WAIT 32768,32,32
```

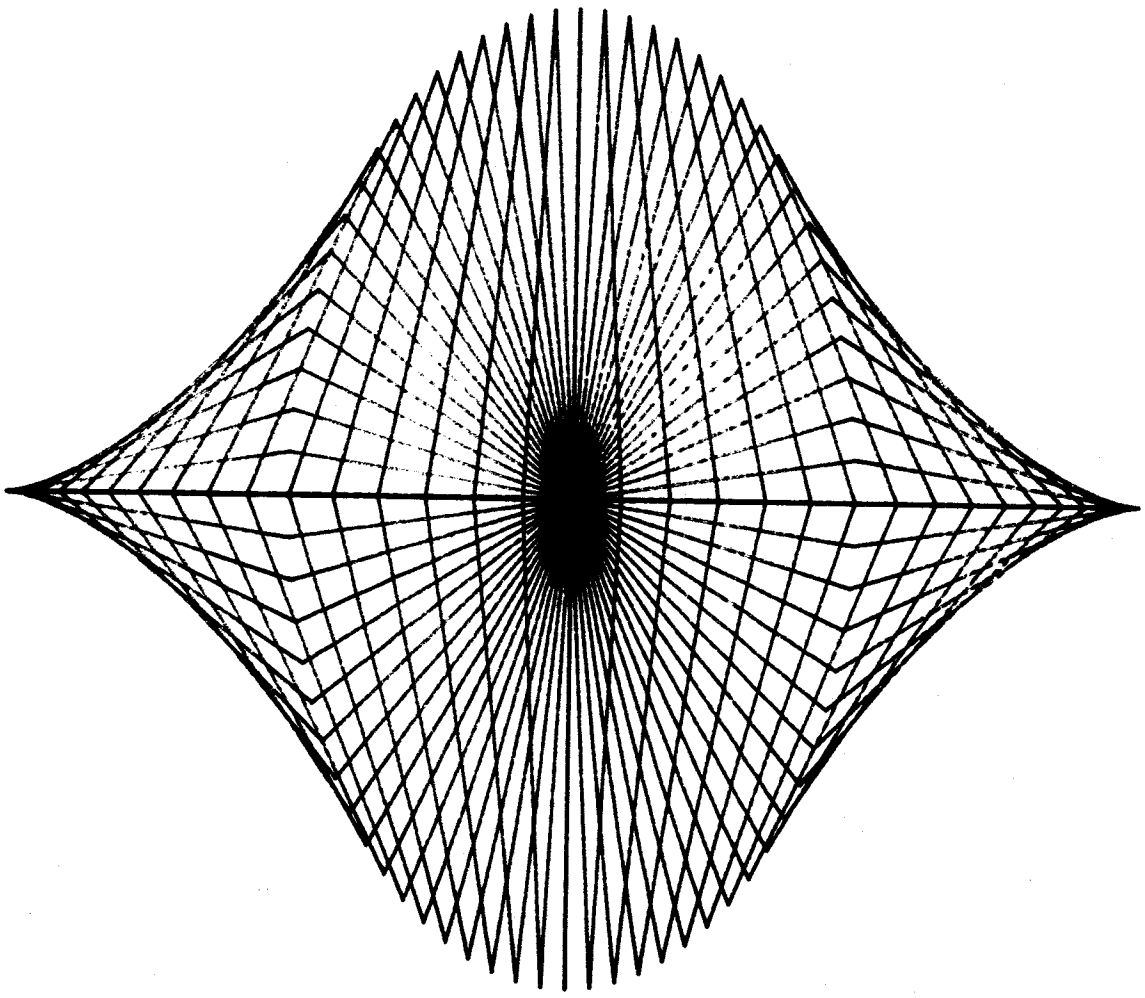
   Experiment with other characters.

If anyone knows of other such peculiar crashes and would like more users to see them, please submit them to The Transactor for publication. - Karl J.

MICRO - AGE '79

The Micro - Age Show at Sheridan College (June 12,13,14) was a big success for all who participated and a pleasure for all who attended. Of the Commodore dealers that participated (House of Computers, Home Computer Centre, Computerland, Compucentre, Nakcomm Sales & Service and BMB Compuscience of Canada) equipment demonstrated included the PET disk and printer with the new PET text editor, a telephone modem tied with the U. of T. main installment and an HP X-Y plotter that produces excellent results (below).

My regards and congratulations to Frank Winter and Margo Martin whose efforts were rewarded by the shows great success. Congratulations also to the speakers for their informative and entertaining seminars.



Merging PET programs: a final report — Jim Butterfield, Toronto

To wrap up the various activities surrounding merging or UNLIST, and bring them up to date with information on new ROM:

I. To change a program into a data file on cassette tape:

Mount blank tape on cassette 1. Type:

```
OPEN 1,1,1 : CMD 1 : LIST
```

Cassette tape will write. When writing is complete, the flashing cursor will return, but PET will not print READY - the word READY is in fact written on tape. Now close the CMD and tape file with:

```
PRINT#1 : CLOSE 1
```

This "merge" tape may now be saved for any future occasion.

Variations:
--the file may be named, e.g., OPEN 1,1,1,"TEST MERGE": ... etc. It's good practice to name files if you plan to keep them.
--if desired you may copy only part of the program to tape, e.g., ... CMD 1 : LIST 500-700 ... This is a handy way to extract subroutines from a larger program.

II. To merge a data file (in the above format) into program space:

The procedure is slightly different on original ROM as compared to the new ROM, which I'll call upgrade ROM.

The program with which you wish to merge must first be loaded into memory. The following procedure may be repeated many times, so that you may merge several program blocks together.

Mount "merge" tape on cassette 1. Type:

```
Original ROM:   POKE 3,1 : OPEN 1
Upgrade ROM:    POKE 14,1: OPEN 1
```

Tape will now be read. Eventually, the computer will report FOUND and the cursor will return.

Now: clear the screen and press exactly three cursor downs. Type:

```
Original ROM:   POKE 611,1 : POKE 525,1 : POKE 527,13 : ?"h"
Upgrade ROM:    POKE 175,1 : POKE 158,1 : POKE 623,13 : ?"h"
```

('h' is the cursor home key - it will print as a reverse S).

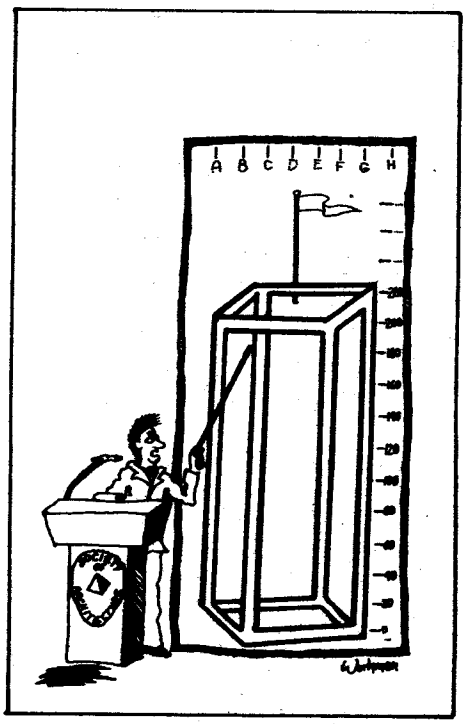
As soon as you press RETURN at the end of this line, the word READY will appear above the line, and tape will move. When the merge is complete, the computer will print either ?OUT OF DATA ERROR or ?SYNTAX ERROR below the line. This is normal and does not signify a real error. The job is now complete.

Note the four new items:

--a new POKE statement before OPEN 1;
--three cursor downs before the final POKE;
--only one final POKE line to be typed;
--no need to close the file at end of merge.

The new system is simpler, and also corrects a minor problem on the original POKE611 merge. Few people spotted it, but the original procedure caused line 1 to disappear.

Jim Butterfield's latest memory map for the New ROMs appears at the back of this bulletin. Thank you Jim!



'Gentlemen, This Is a Structural Analysis of the Proposed Shopping Complex Made by Our Arch/200 Stress Analyzer. Of Course, You May Want a Second Opinion.'

Courtesy Computerworld Newspaper

PET to Teletype Interface

The interface described below was received from Lt. W. Hawes in Nova Scotia. Note: it will operate with 8 level TTY's but not 5 level machines. Thank you Lieutenant Hawes.

Interface Description

The cct. shown in Fig. 1 is a modification of an interface that was originally built in June '78 to output to a TTY from the PET Parallel User Port. The problem with the Parallel port was that software was required to be resident in memory in order to output data and LISTing of programs was not possible since the operating system has control during a LIST. Clearly the way to go was from the IEEE 488 Port.

The modification to output from the IEEE Port was based on the cct. by Prentice Orwell (Jul/Aug 78 Pet User Notes). Some of the features of my original cct, such as UART vice shift register and clock frequency from PET vice interface oscillator, were retained.

My cct. is as shown in Fig. 1 It uses a +5v and -12v (originally only a dual supply UART was immediately available) for both the UART and the 20mA current loop. The cct. could be further simplified to a single +5v supply as shown in Fig. 2 by using a single supply UART such as the AVA - 1014A or equivalent. The 20mA loop could then be constructed using spare inverters on the 4049's.

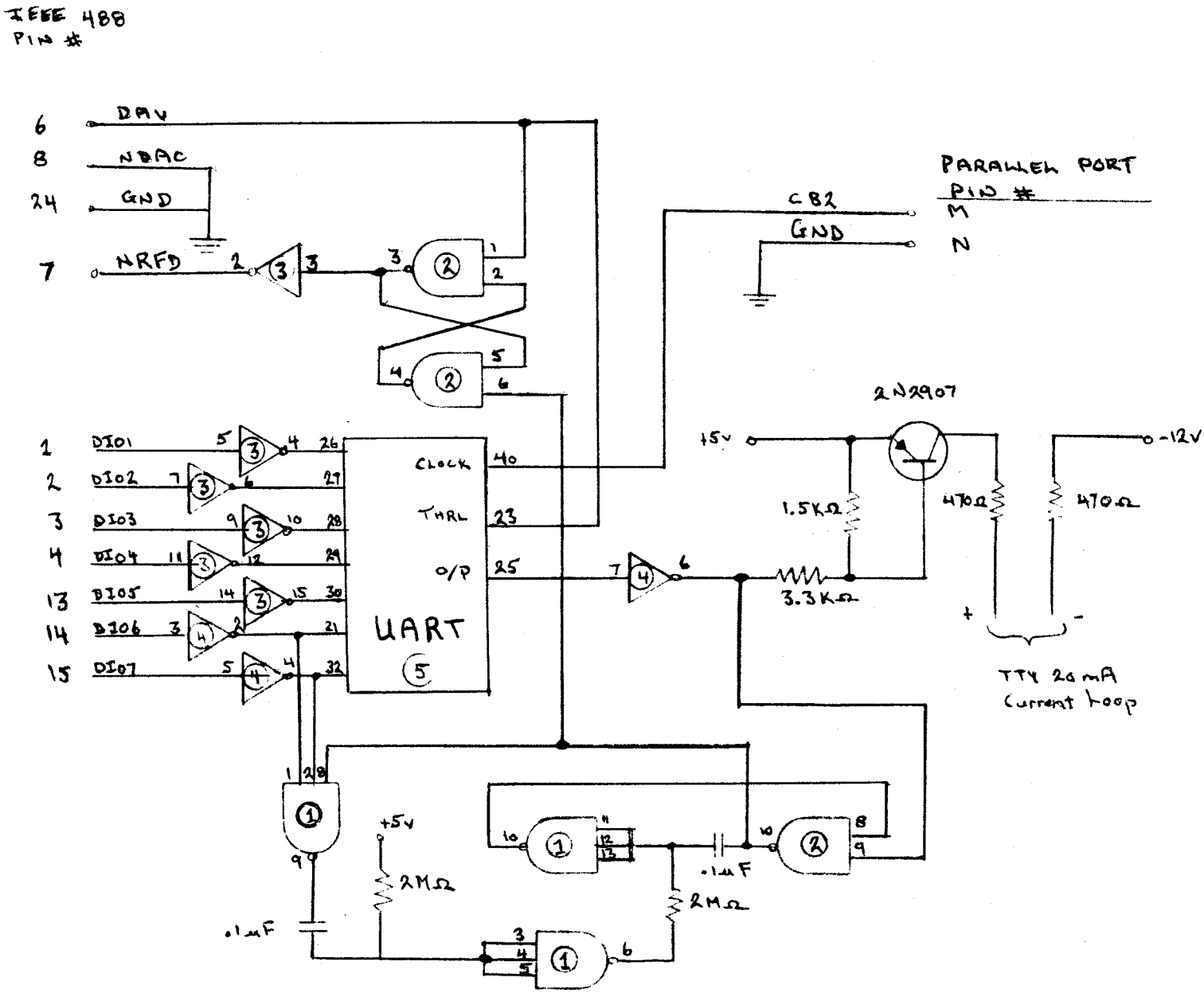
As stated above, hardware is reduced by omitting the interface oscillator. PET itself supplies the 1760 Hz (16 x baud rate) UART clock frequency from CB2 on the parallel port (see Generating Square Waves With The PET by J.R. Kinnard - MAR/APR '78 PET User Notes).

Circuit Operation

Initialize	: POKE 59467,16 : POKE 59464,69 : POKE 59466,51 (outputs 1760 Hz from CB2 to UART, tape I/O disabled)
Operate	: OPEN 4,4 : CMD 4 (Printer primary output device - enter from keyboard to LIST or include in program to be RUN
Return to Screen	: PRINT# 4 (from keyboard or include in program
System Recover	: POKE 59467,0 (restores correct tape I/O)

PET IEEE 488 / TTY 20mA Current Loop

Figure 1

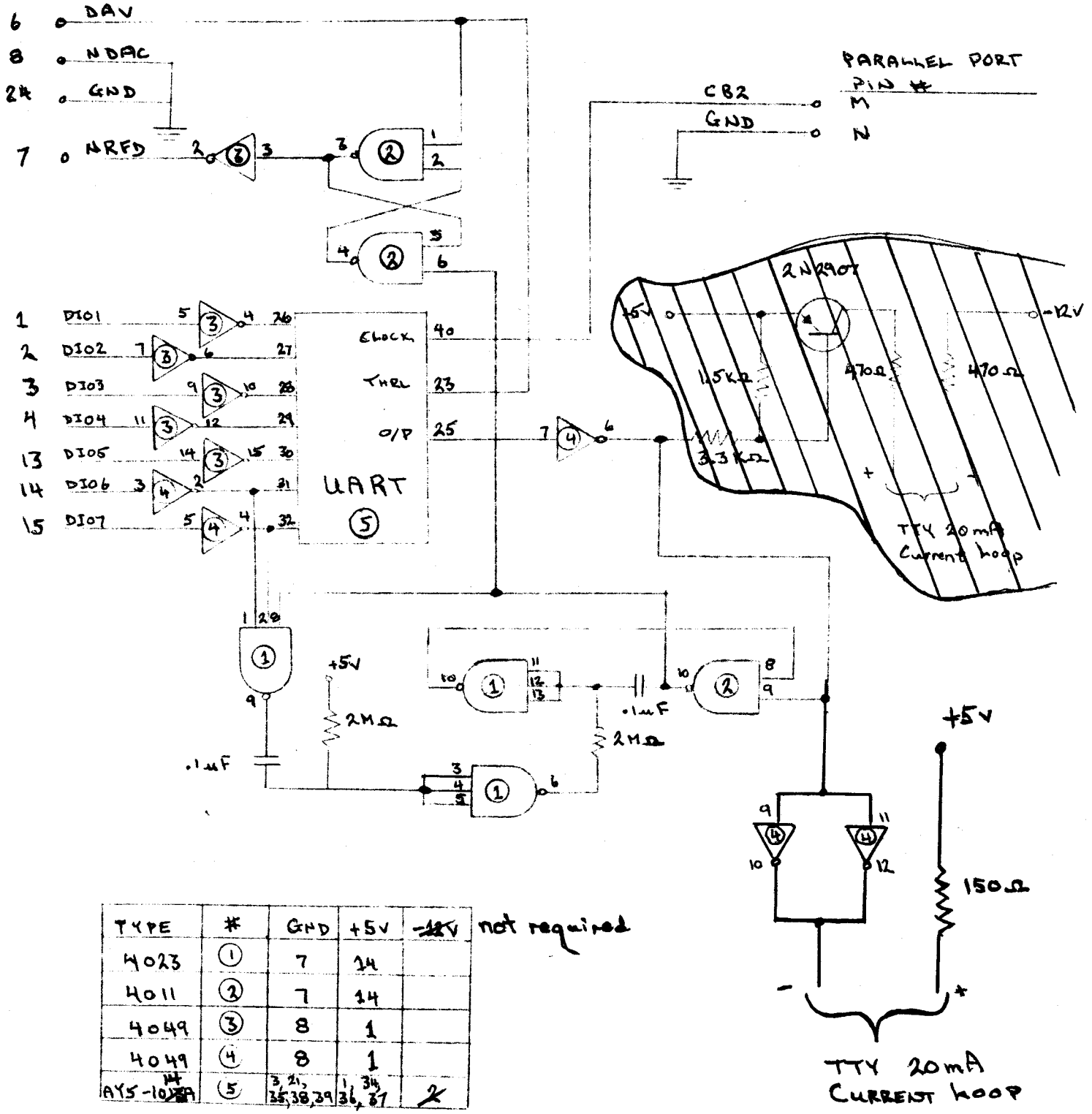


TYPE	#	GND	+5V	-12V
4023	①	7	14	
4011	②	7	14	
4049	③	8	1	
4049	④	8	1	
AY5-1013A	⑤	3, 21, 35, 38, 39	1, 34, 36, 37	2

PET IEEE 488 / TTY 20mA Current Loop

Single +5v Power Supply Option

Figure 2



TYPE	#	GND	+5V	-12V
4023	①	7	14	
4011	②	7	14	
4049	③	8	1	
4049	④	8	1	
AY5-1014A	⑤	3, 21, 35, 38, 39	1, 34, 36, 37	X

not required

Additional I/O Interface

Mr. K. Erler of Edson Alberta writes in with:

...a schematic of an interfacing idea of mine. It simply interfaces a second VIA chip to the PET, thus tripling the user's I/O capability. Most of it is direct interfacing -- all but the address lines which had to be decoded.

The circuit uses only 4 three input 'AND' gates and one buffer inverter. Once assembled, it connects directly on to the Memory Expansion Port - J4.

After connecting it, operation is very simple. The circuit is designed to use the top 16 bytes of RAM expansion space and since most PETs have only 8K (32K at the most) the very top of the memory would not be used.

The addresses are as follows:

32752 - ORB	32760 - T2L-L T2C-L
32753 - ORA	32761 - T2C-H
32754 - DDRB	32762 - SR
32755 - DDRA	32763 - ACR
32756 - T1L-L T1C-L	32764 - PCR
32757 - T1C-H	32765 - IFR
32758 - T1L-L	32766 - IER
32759 - T1L-H	32767 - ORA (no hand shake)

The advantages are that you get not only PA lines, but also the PB lines and CB1 & CA2 lines.

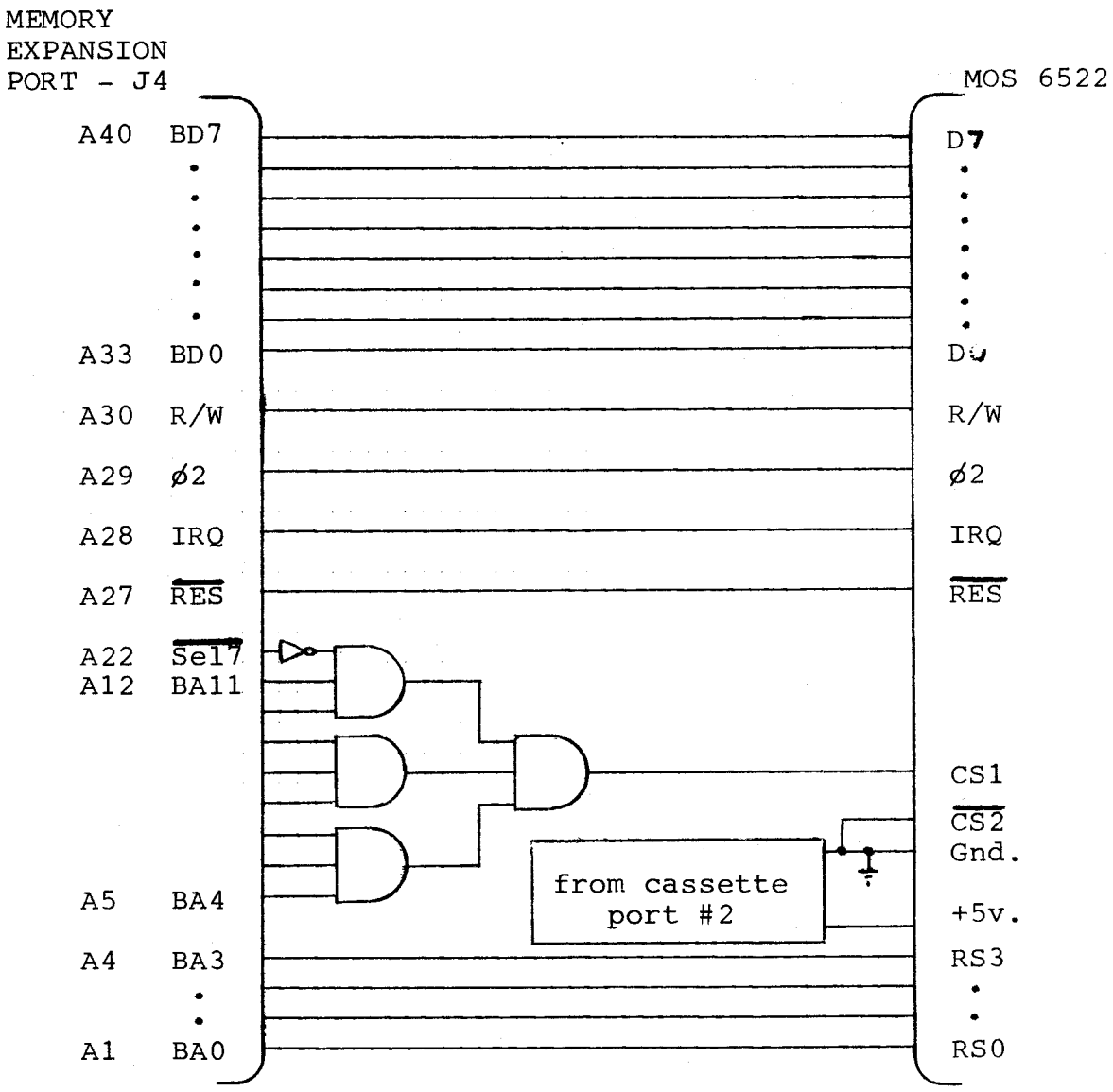
The operation is as with the other VIA - PEEK and POKE, etc, only with the previously listed addresses.

Outut Example

To create a tone on CB2...

```
POKE 32762,15  (SR)
POKE 32760,155 (Timer 2)
POKE 32763,16  (ACR)
```

Great idea, Kevin! Thank you. The schematic follows...



TTY Interface

The Microtronics M65 Morse Flash TTY Interface and Software. It can operate anywhere up to 100 wpm (send and receive modes). Fully compatible with 8, 16, and 32K PETs.

Assembled form - 149.95

Kit form - 109.95

For more information contact:

IRISCO Du QUEBEC
537 Boul Charest Est
Qubec City, Quebec
G1K 3J2

PET Reset Revisited — Bill Twyman

On the last page of the last Transactor there was an ad for a "PET Reset Button" being sold by Gord Reithmeyer. At first glance the normal thought is 'so what - I can do the same thing by turning off my PET.'

Quite true, if you are speaking about the 8K PET. If however you have one of the "New ROM" machines, you will discover a small bonus. As we all know, cutting power or entering reset will kill most, if not all, of BASIC memory.

Now enters the bonus - On PETs equipped with new ROMs, if you hold the Diagnostic Sense (Pin 5 on the Parallel User Port) low, (short to Pin 1) and then push the reset button, you will find yourself in the Machine Language Monitor. Now unshort Pin 5. At this point simply exit the monitor (type X and RETURN), and re-enter the monitor (SYS1024 or SYS4). You will find that you have preserved all of BASIC memory (re-exit monitor and try LIST).

Note 1: It is necessary to exit the monitor and re-enter it at least once in order not to crash again.

Note 2: This procedure will restore full memory on all crashes which:
a) do not involve Ø Page
b) do not in themselves rewrite memory.

For those who are interested, Gord has just completed manufacture of a device which accomplishes all of the above and attaches very neatly to the Parallel User Port. It sells for about $20.00.

Below is the table that was to follow last month's article on the LIST chain and was accidentally omitted. Unfortunately, the article on how PET stores variables could not be completed for this month's bulletin but will appear next month.

0: End of line
1-31: unused
32: space
33: !
34: "
35: #
36: $
37: %
38: &
39: '
40: (
41:)
42: *
43: +
44: ,
45: -
46: .
47: /
48: 0
49: 1
50: 2
51: 3
52: 4
53: 5
54: 6
55: 7
56: 8
57: 9
58: :
59: ;
60: <
61: =
62: >
63: ?
64: @
65: A
66: B
67: C
68: D
69: E
70: F
71: G
72: H
73: I
74: J
75: K
76: L
77: M
78: N
79: O
80: P
81: Q
82: R
83: S
84: T
85: U
86: V
87: W
88: X
89: Y
90: Z
91: [
92: \
93:]
94: ↑
95: ←
96: space
97: !
98: "
99: #
100: $
101: %
102: &
103: '
104: (
105:)
106: *
107: +
108: ,
109: -
110: .
111: /
112: 0
113: 1
114: 2
115: 3
116: 4
117: 5
118: 6
119: 7
120: 8
121: 9
122: :
123: ;
124: <
125: =
126: >
127: ?
128: END
129: FOR
130: NEXT
131: DATA
132: INPUT#
133: INPUT
134: DIM
135: READ
136: LET
137: GOTO
138: RUN
139: IF
140: RESTORE
141: GOSUB
142: RETURN
143: REM
144: STOP
145: ON
146: WAIT
147: LOAD
148: SAVE
149: VERIFY
150: DEF
151: POKE
152: PRINT#
153: PRINT
154: CONT
155: LIST
156: CLR
157: CMD
158: SYS
159: OPEN
160: CLOSE
161: GET
162: NEW
163: TAB(
164: TO
165: FN
166: SPC(
167: THEN
168: NOT
169: STEP
170: +
171: -
172: *
173: /
174: ↑
175: AND
176: OR
177: >
178: =
179: <
180: SGN
181: INT
182: ABS
183: USR
184: FRE
185: POS
186: SQR
187: RND
188: LOG
189: EXP
190: COS
191: SIN
192: TAN
193: ATN
194: PEEK
195: LEN
196: STR$
197: VAL
198: ASC
199: CHR$
200: LEFT$
201: RIGHT$
202: MID$
203-254: unused
255: π

Memory locations for ROM upgrade on PET computers

Jim Butterfield, Toronto

0000-0002	0-2	USR Jump instruction
0003	3	Search character
0004	4	Scan-between-quotes flag
0005	5	Basic input buffer pointer; # subscripts
0006	6	Default DIM flag
0007	7	Type: FF=string, 00=numeric
0008	8	Type: 80=integer, 00=floating point
0009	9	DATA scan flag; LIST quote flag; memory flag
000A	10	Subscript flag; FNx flag
000B	11	0=input; 64=get; 152=read
000C	12	ATN sign flag; comparison evaluation flag
000D	13	input flag; suppress output if negative
000E	14	current I/O device for prompt-suppress
0011-0012	17-18	Basic integer address (for SYS, GOTO etc)
0013	19	Temporary string descriptor stack pointer
0014-0015	20-21	Last temporary string vector
0016-001E	22-30	Stack of descriptors for temporary strings
001F-0020	31-32	Pointer for number transfer
0021-0022	33-34	Misc. number pointer
0023-0027	35-39	Product staging area for multiplication
0028-0029	40-41	Pointer: Start-of-Basic memory
002A-002B	42-43	Pointer: End-of-Basic, Start-of-Variables
002C-002D	44-45	Pointer: End-of-Variables, Start-of-Arrays
002E-002F	46-47	Pointer: End-of-Arrays
0030-0031	48-49	Pointer: Bottom-of-Strings (moving down)
0032-0033	50-51	Utility string pointer
0034-0035	52-53	Pointer: Limit of Basic Memory
0036-0037	54-55	Current Basic line number
0038-0039	56-57	Previous Basic line number
003A-003B	58-59	Pointer to Basic statement (for CONT)
003C-003D	60-61	Line number, current DATA line
003E-003F	62-63	Pointer to current DATA item
0040-0041	64-65	Input vector
0042-0043	66-67	Current variable name
0044-0045	68-69	Current variable address
0046-0047	70-71	Variable pointer for FOR/NEXT
0048	72	Y save register; new-operator save
004A.	74	Comparison symbol accumulator
004B-004C	75-76	Misc numeric work area
004D-0050	77-80	Work area; garbage yardstick
0051-0053	81-83	Jump vector for functions
0054-0058	84-88	Misc numeric storage area
0059-005D	89-93	Misc numeric storage area
005E-0063	94-99	Accumulator#1: E,M,M,M,M,S
0064	100	Series evaluation constant pointer
0065	101	Accumulator hi-order propogation word
0066-006B	102-107	Accumulator#2
006C	108	Sign comparison, primary vs. secondary
006D	109	low-order rounding byte for Acc#1
006E-006F	110-111	Cassette buffer length/Series pointer

Memory map, upgrade ROM, contd.

0070-0087	112-135	Subrtn: Get Basic Char; 77,78=pointer
0088-008C	136-140	RND storage and work area
008D-008F	141-143	Jiffy clock for TI and TI$
0090-0091	144-145	Hardware interrupt vector
0092-0093	146-147	Break interrupt vector
0094-0095	148-149	NMI interrupt vector
0096	150	Status word ST
0097	151	Which key depressed: 255=no key
0098	152	Shift key: 1 if depressed
0099-009A	153-154	Correction clock
009B	155	Keyswitch PIA: STOP and RVS flags
009C	156	Timing constant buffer
009D	157	Load=0, Verify=1
009E	158	# characters in keyboard buffer
009F	159	Screen reverse flag
00A0	160	IEEE-488 output flag: FF=character waiting
00A1	161	End-of-line-for-input pointer
00A3-00A4	163-164	Cursor log (row, column)
00A5	165	IEEE-488 output character buffer
00A6	166	Key image
00A7	167	0=flashing cursor, else no cursor
00A8	168	Countdown for cursor timing
00A9	169	Character under cursor
00AA	170	Cursor blink flag
00AB	171	EOT bit received
00AC	172	Input from screen/input from keyboard
00AD	173	X save flag
00AE	174	How many open files
00AF	175	Input device, normally 0
00B0	176	Output CMD device, normally 3
00B1	177	Tape character parity
00B2	178	Byte received flag
00B4	180	Tape buffer character
00B5	181	Pointer in filename transfer
00B7	183	Serial bit count
00B9	185	Cycle counter
00BA	186	Countdown for tape write
00BB	187	Tape buffer#1 count
00BC	188	Tape buffer#2 count
00BD	189	Write leader count; Read pass1/pass2
00BE	190	Write new byte; Read error flag
00BF	191	Write start bit; Read bit seq error
00C0	192	Pass 1 error log pointer
00C1	193	Pass 2 error correction pointer
00C2	194	0=Scan; 1-15=Count; $40=Load; $80=End
00C3	195	Checksum for read; Leader length for write
00C4-00C5	196-197	Pointer to screen line
00C6	198	Position of cursor on above line

Memory map, upgrade ROM, contd.

00C7-00C8	199-200	Utility pointer: tape buffer, scrolling
00C9-00CA	201-202	Tape end address/end of current program
00CB-00CC	203-204	Tape timing constants
00CD	205	00=direct cursor, else programmed cursor
00CE	206	Timer 1 enabled for tape read; 00=disabled
00CF	207	EOT signal received from tape
00D0	208	Read character error
00D1	209	# characters in file name
00D2	210	Current logical file number
00D3	211	Current secondary addrs, or R/W command
00D4	212	Current device number
00D5	213	Line length (40 or 80) for screen
00D6-00D7	214-215	Start of tape buffer, address
00D8	216	Line where cursor lives
00D9	217	Last key input; buffer checksum; bit buffer
00DA-00DB	218-219	File name pointer
00DC	220	Number of keyboard INSERTs outstanding
00DD	221	Write shift word/Receive input character
00DE	222	#blocks remaining to write/read
00DF	223	Serial word buffer
00E0-00F8	224-248	Screen line table: hi order address & line wrap
00F9	249	Cassette#1 status switch
00FA	250	Cassette#2 status switch
00FB-00FC	251-252	Tape start address
0100-010A	256-266	Binary to ASCII conversion area
0100-013E	256-318	Tape read error log for correction
0100-01FF	256-511	Processor stack area
0200-0250	512-592	Basic input buffer
0251-025A	593-602	Logical file number table
025B-0264	603-612	Device number table
0265-026E	613-622	Secondary address, or R/W cmd, table
026F-0278	623-632	Keyboard input buffer
027A-0339	634-825	Tape#1 buffer
033A-03F9	826-1017	Tape#2 buffer
03FA-03FB	1018-1019	Vector for Machine Language Monitor
0400-7FFF	1024-32767	Available RAM including expansion
8000-8FFF	32768-36863	Video RAM
9000-BFFF	36864-49151	Available ROM expansion area
C000-E0F8	49152-57592	Microsoft Basic interpreter
E0F9-E7FF	57593-59391	Keyboard, Screen, Interrupt programs
E810-E813	59408-59411	PIA1 - Keyboard I/O
E820-E823	59424-59427	PIA2 - IEEE488 I/O
E840-E84F	59456-59471	VIA - I/O and Timers
F000-FFFF	61440-65535	Reset, tape, diagnostic monitor

E810	DIAGNOS. SENSE	IEEE EOI in	CASSETTE SENSE #2 #1	KEYBOARD ROW SELECT PA			59408
E811	TAPE#1 INPUT FLAG	...	SCREEN BLANK OUTPUT (UNUSED ON 32K) CA2	DDRA ACCESS	CASSETTE #1 READ CONTROL CA1		59409
E812	KEYBOARD ROW INPUT						59410
E813	RETRACE I FLAG	...	CASSETTE #1 MOTOR OUTPUT CB2	DDRB ACCESS	RETRACE INTERR. CONTROL CB1		59411

E820	IEEE-INPUT						59424
E821	$\overline{ATN}$ I FLAG	...	IEEE $\overline{NDAC}$ out CB2	DDRA ACCESS	IEEE $\overline{ATN}$ in CONTROL CA1		59425
E822	IEEE-OUTPUT						59426
E823	$\overline{SRQ}$ I FLAG	...	IEEE $\overline{DAV}$ out CB2	DDRB ACCESS	IEEE $\overline{SRQ}$ in CONTROL CB1		59427

E840	$\overline{DAV}$ in	$\overline{NRFD}$ in	RETRACE IN	CASS #2 MOTOR	CASSETTE OUTPUT	$\overline{ATN}$ out	$\overline{NFRD}$ out	$\overline{NDAC}$ in PB	59456
E841									59457
E842	DIRECTION REGISTER B (FOR E840)								59458
E843	DIRECTION REGISTER A (FOR E84F) (P.U.P.)								59459
E844	TIMER 1							L	59460
E845	WRITE							H	59461
E846	TIMER 1							L	59462
E847	LATCH							H	59463
E848	TIMER 2							L	59464
E849								H	59465
E84A	SHIFT REGISTER								59466
E84B	T1 CONTROL PB7 OUT, ONE-SHOT/FREE-RUN		T2 CONTR. PB6 SENSE	SHIFT REG. CONTROL			PB,PA LATCH CONTROL		59467
E84C	CB2 (P.U.P. PWM) CONTROL IN/OUT			CB1 in CASSETTE#2 POLARITY	CA2 (Graphics, Lower Case) Control IN/OUT			CA1 in POLARITY	59468
E84D	IRQ STATUS	T1 INT	T2 INT	CB1 CASS#2 INT	CB2 INT	SR INT	CA1 (P.U.P.B) INT	CA2 INT	59469
E84E	ENABLE CLEAR/SET	T1 INT ENAB	T2 INT ENAB	CB1 INT ENAB	CB2 INT ENAB	SR INT ENAB	CA1 INT ENAB	CA2 INT ENAB	59470
E84F	PARALLEL USER PORT I/O (PA)							PA	59471